ABSTRACT

Title of dissertation: PLANETARY SCALE DATA STORAGE

Benjamin Bengfort
Doctor of Philosophy, 2018

Dissertation directed by: Professor Peter J. Keleher
Department of Computer Science

The success of virtualization and container-based application deployment has fundamentally changed computing infrastructure from dedicated hardware provisioning to on-demand, shared clouds of computational resources. One of the most interesting effects of this shift is the opportunity to localize applications in multiple geographies and support mobile users around the globe. With relatively few steps, an application and its data systems can be deployed and scaled across continents and oceans, leveraging the existing data centers of much larger cloud providers.

The novelty and ease of a global computing context means that we are closer to the advent of an Oceanstore, an Internet-like revolution in personalized, persistent data that securely travels with its users. At a global scale, however, data systems suffer from physical limitations that significantly impact its consistency and performance. Even with modern telecommunications technology, the latency in communication from Brazil to Japan results in noticeable synchronization delays that violate user expectations. Moreover, the required scale of such systems means that failure is routine.

To address these issues, we explore consistency in the implementation of distributed logs, key/value databases and file systems that are replicated across wide areas. At the core of our system is hierarchical consensus, a geographically-distributed consensus algorithm that provides strong consistency, fault tolerance, durability, and adaptability to varying user access patterns. Using hierarchical consensus as a backbone, we further extend our system from data centers to edge regions using federated consistency, an adaptive consistency model that gives satellite replicas high availability at a stronger global consistency than existing weak consistency models.

In a deployment of 135 replicas in 15 geographic regions across 5 continents, we show that our implementation provides high throughput, strong consistency, and resiliency in the face of failure. From our experimental validation, we conclude that planetary-scale data storage systems can be implemented algorithmically without sacrificing consistency or performance.

PLANETARY SCALE DATA STORAGE

by

Benjamin Bengfort

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Peter J. Keleher, Chair/Advisor
Professor Dave Levin
Professor Amol Deshpande
Professor Daniel Abadi
Professor Derek C. Richardson

Preface

If needed.

Foreword

If needed.

Dedication

To Irena and Henry.

Acknowledgments

I could not have done this alone.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

EC Eventual consistency
HC Hierarchical consensus

Chapter 1: Introduction

Ubiquitous computing is becoming the norm. More users are mobile Non-human users (IoT is generating a lot of data) Applications are global Requires transparent persistence

Oceanstore provides a model for untrusted global-scale data Requires connectivity, security, durability (consistency), divorced from location Oceanstore model relies on encryption and async commit with conflict resolution Two tiers for updates: Byzantine quorum and eventual consistency layer

Cloud revolution The model has changed: client-server interactions and siloed data (not files) Trusted replicas are available across the globe (or secured with encryption/SUNDR) Updates are still too fast, however, centralization means that consistency is now the number one issue. FIGURE: locations of cloud computing data centers

Toward Edge and Fog computing Data systems are becoming increasingly complex (many publishers, few subscribers, distributed accesses) Sensor networks, energy platforms, self-driving vehicle networks Greater localization is required

Our vision: planetary-scale data storage First tier: strong, resilient, fast consensus with hierarchical consensus Second tier: highly available eventual consistency

federated with strong consistency Adaptiveness: online monitoring and network adaptation

Contributions: Design, implementation, and evaluation of hierarchical consensus Design, implementation, and evaluation of federated consistency Design, implementation, and evaluation of bandit-based anti-entropy Design and implementation of planetary-scale key/value store Design and implementation of a planetary-scale file system Description and discussion of distributed systems in the very wide area Description and discussion of dependent scaling of consensus algorithms

Organization of the rest of the paper

Chapter 2: Challenges and Motivations

Is a planetary-scale data system really necessary?

What challenges to these systems face?

What does consistency mean in this model?

2.1 Building Geo-Replicated Services: Case Studies

What they didn't do and what we do better.

BigTable & Spanner

S3

Aurora & Cockroach DB

Approaches: sharding, independent objects, buckets, slow reads

2.2 Requirements for Data Systems

Failure is common Disk failure/replica failure (ODS) Network failure (when repaired, replica comes back online) Unreliability: messages have highly variable latency, out of order messaging Partitions, part of the system cannot speak to the rest of the system

In geo-systems large latency is not the issue! There is a physical limit to

message traffic Writes must be applied in order, reads can reason about staleness Access patterns are also location-dependent

Durability Normally 3 disk replication ensures 2 failures We need to ensure zone+1 disk failures (re Aurora) User-specific data should be accessible everywhere

Fault-Tolerance & availability System should still be available if nodes fail Should be available if zones fail (e.g. hurricane or disaster) at higher latency System should be available at lower consistency if even one replica is available

Adaptability Should respond to changes in user access patterns Should be able to add and remove nodes from the system Should scale with more regions and more replicas

2.3 System Architecture

Describe the architecture of tier 1: HC, tier 2: Federated Fog

Describe the consistency guarantees we claim

Base application is a key/value store

File-system is built upon the key/value store as an object FS

Chapter 3: Hierarchical Consensus

A strong consistency foundation is required - usually implemented with state machine consensus

Paxos/Raft/ePaxos don't scale

Sharding and tablets don't allow for inter-object dependence and the point of failure is the management quorum, which is also not geo-replicated.

HC extends and implements Vertical Paxos such that there is an intersection between quorums that manage objects and a root quorum The result is a grid consistency

3.1 Overview

Root quorum manages object space

Sub quorums manage accesses

Hot spares

3.2 Consensus

3.2.1 Root Consensus

3.2.2 Delegation

3.2.3 Epoch Transitions

- tagspace (prefix-trie) - fuzzy transitions - anti-entropy and tombstones

3.2.4 Subquorum Consensus

3.3 Client Operations

- Sessions - Connect to closest available replica, redirected to closest available leader.

3.3.1 Consistency

3.3.2 Remote Writes

3.3.3 Read Leases

3.4 Fault Tolerance

3.4.1 Failures

3.4.2 Obligations Timeout

3.4.3 The Nuclear Option

3.5 performance Evaluation

Throughput for different cluster sizes

Throughput of Alia vs. Raft

Client cumulative latency distribution

Sawtooth graph

Repartition

Chapter 4: Federated Consistency

Hybrid consistency model

4.1 Overview

4.2 Eventual Consistency

Read/Write Quorums

Anti-Entropy Sessions and Synchronization

Policies: latest writer wins

Bilateral anti-entropy

4.2.1 Consistency Failures

Forks

Stale Reads

4.3 Integration

4.3.1 Communication Integration

4.3.2 Consistency Integration

- Forte Number

4.4 Performance Evaluation

Communication Topology Inconsistencies due to outages Inconsistency due to system latency

Chapter 5: System Implementation

Given its grandiose title, it may seem that the engineering behind the development of a planetary scale data storage system would require thousands of man-hours of professional software engineers and a highly structured development process. In fact, this is not necessarily the case for two reasons. First, data systems benefit from an existing global network topology and commercial frameworks for deploying applications. This means that both the foundation and motivation for creating large geo-replicated systems exists, as described earlier. Second, like the internet, complex global systems emerge through the composition of many simpler components following straight forward rules [1]. Instead of architecting a monolithic system, the design process is decomposed to reasoning about the behavior of single processes. Rather than being built, a robust planetary data system evolves from its network environment.

To facilitate system evolution, the consistency models we have described thus far have been *composable* to allow heterogeneous replicas to participate in the same system. However, while composability allows us to reason about consistency expectations, it does not necessarily mean interoperability. In order to ensure reason about system expectations we must outline our assumptions for communication, se-

curity, processing, and data storage. In this chapter, we describe the implementation of the replicas and applications of our experimental system and the assumptions we made.

5.1 System Model

A *replica* is an independent process that maintains a portion of the objects stored as well as a *view* of the state of the entire system. Replicas must be able to communicate with one another and may also *serve* requests from clients. A system is composed of multiple communicating replicas and is defined by the behavior the replicas. For example, a totally replicated system is one where each replica stores a complete copy of all objects as in the primary-backup approach [2], whereas a partially replicated system ensures durability such that multiple replicas store the same object but not all replicas store all objects as in the Google File System [3]. At the scale of a multi-region, globally deployed system, we assume that total replication is impractical and primarily consider the partial replication case.

Let's unpack some of the assumptions made by the seemingly simple statements made in the previous paragraph. First, independence means replicas have a shared-nothing architecture [4] and cannot share either memory or disk space. For practical purposes of fault tolerance, we generally assume that there is a one-to-one relationship between a replica and a disk so that a disk failure means only a single replica failure. Second, that each replica must maintain a view of the state of the entire system means both that replicas must be aware of their peers on the network

and that they should know the locations of objects stored on the network. A strict interpretation of this requirement would necessarily make system membership brittle as it would be difficult to add or remove replicas. Alternatively, a centralized interpretation of the view requirement would allow for

Second, the ability to communicate with replicas and serve requests from clients means that the replica must be addressable.

This requirement is seemingly innocuous when taken by itself, however when we also describe a replica as requiring a view of the state of the entire system, it means that a replica must know about the existence of all other replicas in the system. A strict interpretation of having a complete view would necessarily make the system composition brittle, unable to add or remove replicas. Instead we take a less strict view

- networking - actors - event loop - reasons why the above are important -

5.2 Applications

5.2.1 Distributed Log

5.2.2 Key-Value Database

5.2.3 File System

5.3 Consistency Model

Client-side vs. system-side consistency

Log model of consistency

Continuous consistency scale

Grid consistency model

5.4 Conclusion

We did not optimize our research for the minimum set of assumptions required to facilitate interoperability between heterogeneous replicas. However, we hope that the assumptions we did make shed light on what is required to achieve the minimum set of assumptions.

Further research is required to ...

Chapter 6: Adaptive Consistency

Monitor and Optimize.

Replica placement, object placement

6.1 Bandit-Based Approaches

6.1.1 Rewards Function

6.2 Access Temperature Approaches

- Expected model of access patterns (daylight).

Chapter 7: Related Work

Chapter 7: Conclusion

Appendix A: Formal Specification

Will add formal specification here.

Bibliography

[1] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the Internet. 39(5):22–31.

[2] Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. The primary-backup approach. 2:199–216.

[3] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43. ACM.

[4] Michael Stonebraker. The case for shared nothing. 9(1):4–9.